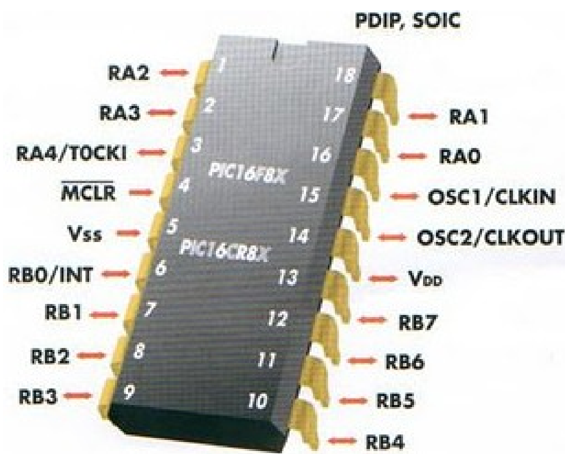
المستوى : 3 ت ر	**الموضوع : وظيفة التحكم**	
الشعبة : هــ ك	**الدارات المنطقية المبرمجة على شكل دارة مندمجة (الميكرومراقب: PIC16f84A)**	العام الدراسي : 11/12

الإشكالية :

يمكن إنجاز الوظائف الحسابية و المنطقية بواسطة دارات مندمجة تماثلية أو منطقية. لكن عندما يصبح العنصر التقني معقد و يتطلب إنجاز معالجة معلومات كثيرة و بسرعة، يصبح من الضروري استعمال عناصر تتميز بأكثر قدرة و جاهزية للتعامل مع هذه المتطلبات، لهذا نستعمل الميكرومراقب.
بالإضافة إلى ما سبق فإن التكنولوجية المربوطة بدأت شيء فشيء بالاندثار من المحيط الصناعي.

تعريف:

الميكرومراقب هو جيل جديد و مطور من الميكرومعالج حيث أن جميع ملحقات المعالج (ذاكرات، سجلات، الخ...) تم وضعها في شريحة واحدة ومن هنا بناء دارة تحتوي على ميكرومراقب تكون بسيطة وصغيرة وليست معقدة.
العنصر المطروح قيد الدراسة خاص بالشركة **MICROCHIP** ألا و هو: **PIC16F84A**.
الكلمة **PIC** هي علامة سجلة من ابتكار **MICROCHIP، 16** تعني أن العنصر من العائلة الوسطى (-Mid range:14bit) **F**: الذاكرة المستعملة من نوع Flash. **A84**: هو الرقم التعريفي للعنصر.



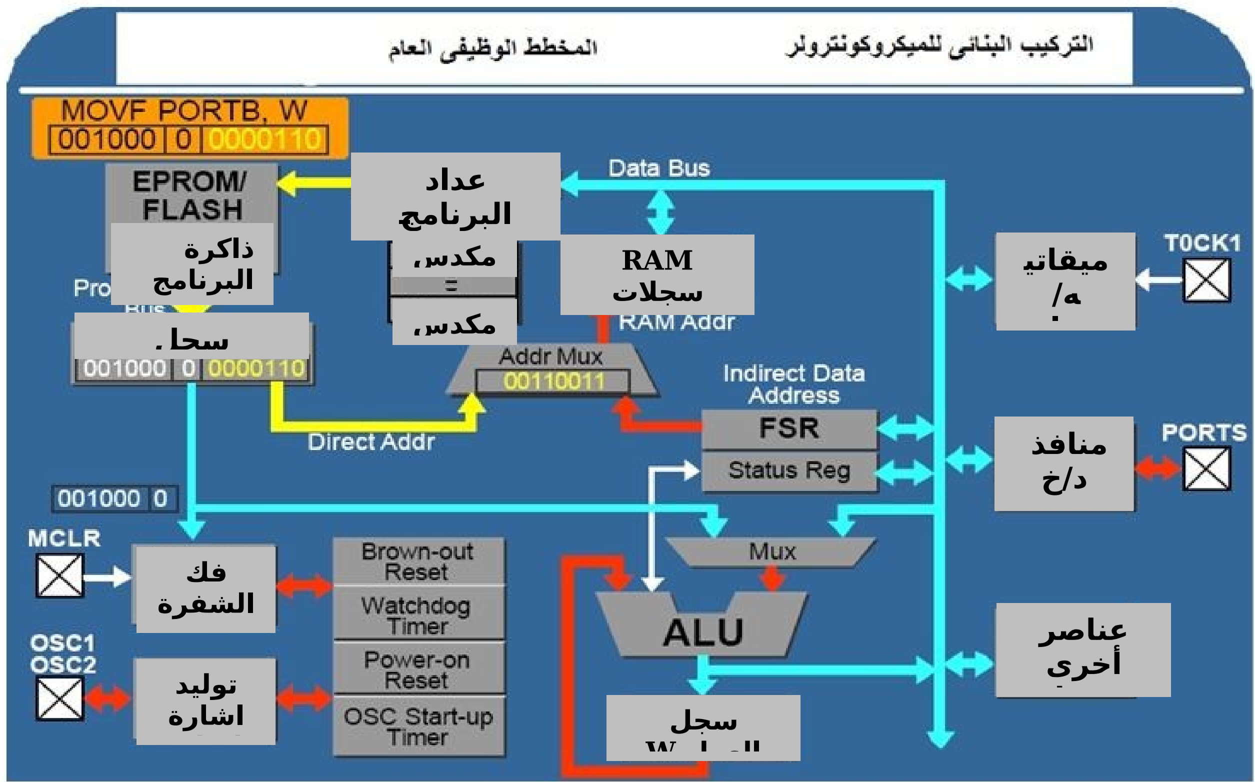
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1

الهيكلة القاعدية للميكرومراقب PIC16F84A:

1-1 تحديد أرجل الميكرومراقب PIC16F84A:

- الأرجل: **RA0, RA1, RA2, RA3, & RA4** عبرة عن منافذ **PORTA.**
- الرجل: **MCLR** مدخل يستعمل من أجل إعادة التشغيل: **RESET** .
- **V_{SS}** و **V_{DD}** هما على الترتيب رجلي التغذية (5V) و الكتلة.
- **OSC1/CLKIN و OSC2/CLKOUT** هما عبارة عن رجلي توصيل دارة المذبذب الخارجي (كريستال أو خلية RC).
- الأرجل: **RB0/INT, RB1,………, RB7** عبرة عن منافذ **PORTB**، كم يمكن استعمال المنفذ **RB0/INT** كمدخل للمقاطعة (Interruption).

الشكل التالي يبين المخطط النموذجي للمكونات الأساسية الخاص بالعائلة الوسطى PIC16 (Mid-range) .



يتكون الميكرومراقب أساسا من العناصر التالية:

- الوحدة الحسابية و المنطقية : 8 bit **UAL**.
- سجل العمل : 8 bit **W**.
- ذاكرة البرنامج من نوع: 1k mots **FLASH** de 14 bit.
- عداد البرنامج: 14 bit.
- مكدس ذو 8 مستويات.
- سجل التعليمة: 14 bit.
- ذاكرة: **RAM** تحتوي على السجلات الخاصة (SFR) بالإضافة إلى موقع به 68 octet للاستعمال العام ، و هي منفصلة إلى جزئين (bank**0** و bank**1**).
- ذاكرة: (64 octet) **EEPROM** ، للكتابة و القراءة. حيث أنها لا تفقد محتواها بانقطاع التغذية فهي جد مفيدة من أجل الاحتفاظ بالمعطيات الدائمة.
- ميقاتية /عداد ذو 8 bit.
- منافذ دخول أو خروج (Port**A** , Port**B**).
- إشارة ساعة داخلية.

1-2 عداد البرنامج و المكدس و ذاكرة البرنامج :

يقوم عداد البرنامج (**program counter**) بعنونة حيز ذاكرة البرنامج سواء أكانت ذاكرة**EPROM** أو . **Flash/EEPROM** و عداد البرنامج في هذه العائلة هو سجل طوله **13 bits**.

يتم تحميل ذاكرة البرنامج بكود (شفرة) البرنامج المطلوب من الميكرومراقب تنفيذه. البرنامج يكون على شكل قائمة من التعليمات ويقوم عداد البرنامج بحفظ عنوان التعليمة التالية التي عليها الدور في التنفيذ أي أنه عمل كمؤشر لذاكرة البرنامج ويمكن لقيمة عداد البرنامج أن تنتقل إلى المكدس ويحدث ذلك عند تنفيذ برنامج فرعي أو حدوث مقاطعة كما في تعليمات CALL وRETURN وRETFIE و RETLW. عداد البرنامج ذو 13-bit ومن ثم فإن الميكرومراقب يمكنه نظريا العنونة من 0000h إلى 1FFFh .

أول موقع في ذاكرة البرنامج يسمى *reset vector* (متجه أو شعاع البداية). عند بدء تشغيل البرنامج لأول مرة (عند توصيل التغذية مثلا) فإن عداد البرنامج يوضع على الصفر (تصفير (0000 و بالتالي فإن أول موقع سوف يشير إليه هو (reset vector متجه البداية). و من ثم فعلى المبرمج أن يضع أول تعليمة له في هذا الموقع.

2- تنظيم ذاكرة الميكرومراقب PIC 16F84A :

2-1 ذاكرة البرنامج FLASH :

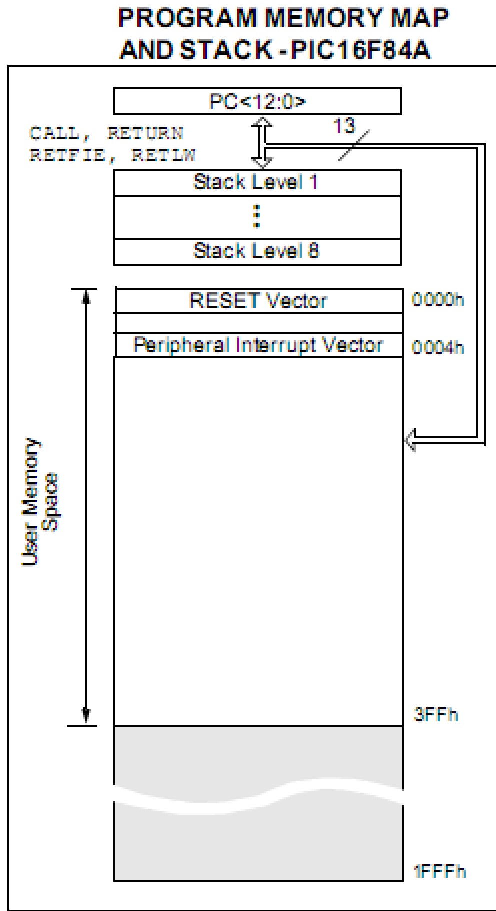
تحتوي على أوامر وتعليمات البرنامج وأيضا مؤشر الوضع في الصفر و مؤشر المقاطعة **INTERRUPTION .**

في الحالات التي يكون فيها:

- **RESET** ⇐ الـميكرومراقب يبدأ من جديد من الموضع **0000h**

و يسمى **مؤشر الوضع في الصفر.**

- **INTERRUPTION** ⇐ الـميكرومراقب يذهب إلى الموضع **0004h** و يسمى **مؤشر المقاطعة.**



2-2 الذاكرة RAM :

تحتوي على 68 بايت للمعطيات وعلى جميع السجلات اللازمة لتسيير النظام و استعمال الملحقات الداخلية والمنافذ.

و كما نلاحظ فهي مرتبة على شكل منطقتين (Bank**0** و Bank **1**).

تتكون الجهة العلوية من 24 موقع (ذو 8 bit) خاص بالسجلات الخاصة، 12 منها في صفحة BANK 0 ذات العناوين من 00h إلى 0Bh و 12 في صفحة BANK 1 ذات العناوين من 80h إلى 8Bh.

كما تتكون الجهة السفلية من 68 خانة معنونة من 0Ch إلى 4Fh وتدعى سجلات الاستعمال العام، و هي مندمجة في الصفحتين.

المساحة المتبقية باللون الرمادي غير مستعملة.

REGISTER FILE MAP - PIC16F84A

File Address	البنك0	البنك1	File Address
00h	Indirect addr.(1)	Indirect addr.(1)	80h
01h	TMR0	OPTION_REG	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	—	—	87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2(1)	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch … 4Fh	68 General Purpose Registers (SRAM)	Mapped (accesses) in Bank 0	8Ch … CFh
50h … 7Fh			D0h … FFh
	Bank 0	Bank 1	

☐ Unimplemented data memory location, read as '0'.

Note 1: Not a physical register.

2-3 ذاكرة المعطيات :

وهي من نوع EEPROM ذات 64 بايت معنونة من 00 إلى 3F تخزن بها المعطيات التي لا يراد فقدانها بانقطاع التغذية.

3- أهم السجلات الأساسية.

3-1 سجل الحالة (Status Register):

STATUS

(03h ; 83h) العنوان

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

- البت 7 و 6 (**IRP**و**RP1**) مثبتان في 0.
- البت 5 (**RP0**) يستعمل للانتقال من و إلى الكومتين 0 و 1 (Bank1 ; Bank0).
- البت 4 (**T0**): خانة المهلة الزمنية و =1 عند توصيل التغذية أو عن طريق التعليمتين CLRWDT و SLEEP .
- البت 3 (**PD**) : خانة انخفاض الطاقة و =1 عند توصيل التغذية أو عن طريق التعليمة CLRWDT .
- البت 2، و =1 عندما تكون نتيجة عملية حسابية أو منطقية تساوي 0، و = 0 إذا لم تكن كذلك.
- البت 0 (**C**) خانة الاحتفظ و =1 إذا نتج تجاوز للخانة الأكثر أهمية (8) عند إجراء عمليات الجمع و الطرح و =0 عدا ذلك.
- البت 1(**DC**) خانة نصف الاحتفاظ و =1 إذا نتج تجاوز للخانة 4 عند إجراء عمليات الجمع و الطرح و = 0 عدا ذلك.

3-2 سجل التهيئة (TRISA) و سجل المنفذ (PORTA):

دور السجل **TRISA** هو إعداد اتجاه كل بت (خط) الخاص بالمنفذ **A (PORTA)** ، بحيث يبرمج كل بت كمدخل إذا أعطي القيمة 1 و مخرج إذا أعطي القيمة 0.

أمّا السجل **PORTA** فكل بت خاص به يمثل صورة المنفذ RAx، كما هو موضح في الشكل الموالي:

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
05h	PORTA	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

Legend: x = unknown, u = unchanged, – = unimplemented, read as '0'. Shaded cells are unimplemented, read as '0'.

ملاحظة: كل من هذان السجلان يحتوي عل 5 بت فقط.

3-3 سجل التهيئة (TRISB) و سجل المنفذ (PORTB):

دور السجل **TRISB** هو إعداد اتجاه كل بت (خط) الخاص بالمنفذ **B (PORTB)** ، بحيث يبرمج كل بت كمدخل إذا أعطي القيمة 1 و كمخرج إذا أعطي القيمة 0.

أمّا السجل **PORTB** فكل بت خاص به يمثل صورة المنفذ **RBx**، كما هو موضح في الشكل الموالي:

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111

يمكن استعمال البت 0 كمدخل لتنفيذ مقاطعة خارجية (**INT**).

3-4 سجل العمل (W Register):

يستعمل هذا السجل كوسيط يلجأ إليه في معظم إجراءات البرمجة.

3-5 سجل الإعدادات (Configuration Register):

موقع سجل الإعدادت ب العنوان 2007h من ذاكرة البرنامج خيث يتم تحديد هذه العدادات عبر البرمجة كما سنرى ذلك لاحقا.

PIC16F84A CONFIGURATION WORD

R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u
CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	PWRTE	WDTE	F0SC1	F0SC0

bit13 bit0

bit 1-0 **FOSC1:FOSC0**: Oscillator Selection bits
11 = RC oscillator 01 = XT oscillator
10 = HS oscillator 00 = LP oscillator

bit 2 **WDTE**: Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

bit 3 **PWRTE**: Power-up Timer Enable bit
1 = Power-up Timer is disabled
0 = Power-up Timer is enabled

bit 13-4 **CP:** Code Protection bit
1 = Code protection disabled
0 = All program memory is code protected

-4 أساسيات البرمجة بلغة الأسمبلر (المجمع)

-1 مشكلة البرمجة والتوافق (إيجاد الحل الوسط)

مشكلة البرمجة ملخصة على النحو التالي :

نحن كبشر نعبر عن أفكارنا بطريقة معقدة وغالبا ما يكون تعريفها بأشكال لغوية غير محددة المعالم. الميكرومراقب يقرأ ويفهم النظام الثنائي ويستجيب بشكل دقيق لتعليمات دقيقة . فهو منطقي إلى أقصى الحدود ويفعل تماما ما يؤمر به. مع هذا الفرق اللغوي كيف يمكن للمبرمج كتابة برامج الميكرومراقب ؟
هناك ثلاثة طرق تفرض نفسها لسد هذه الفجوة و هي:

-1 أن يتعلم الإنسان لغة الآلة .

و هذا ما كان يستخدمه المبرمجين فيما مضى.
بمشقة يكتب كل تعليمة بشفرة الكود الثنائية حتى يستطيع الميكرومراقب من قراءتها .
وهذا بطيء للغاية وممل و عرضة للأخطاء ولكن على الأقل فان المبرمج يستجيب مباشرة لحاجات وقدرات الميكرومراقب.

-2 استخدام لغات البرمجة ذات المستوى المرتفع.

وهذا كما لو كنا بطريقة ما نطلب من الميكرومراقب (الكومبيوتر) تعلم لغتنا .
في لغات البرمجة ذات المستوى المرتفع تكتب التعليمات بشكل يرتبط بإدراكنا للغتنا. في هذه الحالة نحتاج إلى برنامج آخر وهو إما مجمع compiler أو مترجم) interpreter يقوم بتحويل برنامجنا إلى كود لغة الآلة الذي يفهمها الميكرومراقب.
بهذه الطريقة يستمتع المبرمج بوقته في البرمجة كما يكتب برامج متطورة .والنتيجة إن المبرمج ينفصل عن إمكانيات الميكرومراقب وقد يؤدي إلى برنامج غير فعال نسبيا من حيث استخدام الذاكرة وسرعة التنفيذ .

-3 استخدام لغة الأسمبلر (المجمع). و هذا هو الوضع الوسط.

حيث كل تعليمة من مجموعة التعليمات تعطى تذكير (مفكرة =mnemonic) تعبر بصفة مختصرة عن مدلول التعليمة.
عندئذ يكتب المبرمج البرنامج مستخدما التعليمات بالمفكرات .
يجب على المبرمج التفكير بمستوى الميكرومراقب كما لو كان يعمل مباشرة بتعليماته ولكن على الأقل فإن المبرمج لديه مفكرات mnemonics لاستخدامها أفضل من التعامل بشفرة لغة الآلة.

قبل الشروع في البرمجة يجب أولا معرفة التعليمات الخاصة بالميكرومراقب 16F84A و هي بعدد 35 تعليمة.

Mnemonic,	Operands	Description	Cycles	14-Bit Opcode				Status Affected
				MSb			LSb	
BYTE-ORIENTED FILE REGISTER OPERATIONS								
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z
DECFSZ	f, d	Decrement f, Skip if 0	1 (2)	00	1011	dfff	ffff	
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z
INCFSZ	f, d	Increment f, Skip if 0	1 (2)	00	1111	dfff	ffff	
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z
MOVWF	f	Move W to f	1	00	0000	lfff	ffff	
NOP	-	No Operation	1	00	0000	0xx0	0000	
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff	
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z
BIT-ORIENTED FILE REGISTER OPERATIONS								
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff	
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff	
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff	
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff	
LITERAL AND CONTROL OPERATIONS								
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk	
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk	
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk	
RETFIE	-	Return from interrupt	2	00	0000	0000	1001	
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk	
RETURN	-	Return from Subroutine	2	00	0000	0000	1000	
SLEEP	-	Go into standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z

من خلال وثائق الصانع MICROCHIP نلاحظ أن تعليمات الميكرومراقب PIC16F84A تنقسم إلى 3 أقسام:

- **Byte-oriented** التعليمات الموجهة مع محتوى ملف السجل **F** بكامله (بمعنى الخانات الثمانية للسجل byte=8 bit).
- **Bit-oriented** التعليمات الموجهة مع محتوى خانة واحدة لملف السجل **F** (قراءة أو كتابة لأحد البت من بين 8 بت).
- **Literal and Control** التعليمات على شكل كلمة مباشرة (**k**) متكونة من 8 بت، بالإضافة إلى تعليمات التحكم.

نتحول الآن إلى مجموعة التعليمات للعائلة PIC 16 أن نرى أن الجدول مقسم إلى ستة أعمدة وكل تعليمة من ال **35 تعليمة** توجد في سطر واحد .

العمود الأول:
يعطى التعليمة بشكلها الفعلي (سميناه شكل مفكرة(mnemonic ومعها الشفرة (الكود) التي تحدد نوع المعامل الذي سنعمل عليه.(يسمى operand).
يوجد أربع أنواع من شفرات هذا المعامل و هي:

- **المعامل f** أي ملف file وتعني أي موقع في ذاكرة البيانات وهو عدد مكون من 7 خانات (يمكن أن نكوّن من السبع خانات 128 عدد) أي يكون سجل ملف من إجمالي 128 سجل-ملف .
- **المعامل b** أي خانة bit والمطلوب البحث عنها في داخل ملف محدد وهو عدد من 3 خانات (يمكن أن نكون من 3 خانات 8 أعداد، أي تكون خانة من إجمالي 8 خانات) .
- **المعامل d** ويعنى الهدف (أو المقصود) وهو خانة واحدة فقط (bit-1) (يمكن أن نكون من خانة واحدة عددين فهي إما أن تكون 0 أو 1 .
- **المعامل k** ويعنى قيمة ثابتة محددة حرفيا literal وهو عدد مكون من 8خانات إذا كان يعبر عن بيانات ومن 11 خانة إذا كان يعبر عن عنوان.

العمود الثاني :يلخص عمل (وصف لوظيفة) التعليمة.

العمود الثالث :
يبين عدد دورات التعليمة التي تأخذها التعليمة في التنفيذ .

العمود الرابع :
يعطى شفرة (كود) العملية و المسمى opcode لكل تعليمة والمكون من 14خانة وهذا هو الكود الذي ينتجه برنامج الأسمبلر عندما يحول البرنامج من شفرة لغة المجمع إلى شفرة لغة الآلة و من المفيد ملاحظة شفرات المعاملات operand المذكورة بعالية وكيفية تضمينها في كود العملية .

العمود الخامس :
يبين أي الخانات في سجل الحالة Status ستتأثر بالتعليمة .

لنتناول الآن بعض التعليمات من وثائق الصانع على سبيل المثال و التي سنستغلها في برمجة الميكرومراقب:

ملاحظة :
البرمجة بلغة المجمع ليست حساسة لنوع الحروف (كبير أم صغير).
التعليمة : **clrw**
هذه التعليمة تجعل قيمة السجل **W** صفر (ومعنى شكل المفكرة mnemonic : أن **clrw** تذكرنا بكلمة **clear** أي محو أو مسح و الحرف **W** يذكرنا بالسجل **W** وهذا النوع من التعليمات لا يتم تحديد معامل بها.
و العمود الخامس يخبرنا بأن خانة الصفر Z في سجل الحالة سوف تتأثر بالتعليمة و نظرا لأن نتيجة هذه التعليمة دائماصفر فان **Z** دائما تكون **1** : . لا تتأثر أية خانة أخرى من سجل الحالة .

التعليمة: **clrf f**
هذه التعليمة تمسح (تصفر) القيمة الموجودة في موقع من مواقع الذاكرة و الذي نرمز له بالرمز **f** , المبرمج هو الذي يحدد قيمة **f**.
مرة ثانية لأن النتيجة هي صفر فإن خانة الصفر **Z** في سجل الحالة ستتأثر كما سبق .

التعليمة : **addwf f,d**
هذه التعليمة تضيف (تجمع) محتويات السجل **W** إلى محتويات موقع الذاكرة الذي يرمز له بالحرف **f** , و المبرمج هو الذي يحدد قيمة **f**.
هناك اختيار في مكان وضع النتيجة وهذا يتحدد بقيمة خانة المعامل **d** الهدف.
و نظرا لأن النتيجة قد تأخذ قيم مختلفة (كنتيجة لعملية الجمع) فإن جميع خانات شفرة الحالة الثلاثة بسجل الحالة سوف تتاثر بالتعليمة) خانة الصفر **Z** وخانة الحمل **C** وخانة نصف الحمل **DC**).

التعليمة : bcf f,b
هذه التعليمة تمسح (تصفر) خانة واحدة bit في موقع ذاكرة وكل من الخانة والموقع يحدد بمعرفة المبرمج .
رقم الخانة b ياخذ قيمةمن 0 الى 7 لتحديد خانة من الثماني خانات في موقع الذاكرة .
لا يتاثر أي علم من أعلام سجل الحالة حتى لو تخيلنا أن نتيجة التعليمة تجعل موقع الذاكرة صفرا .

التعليمة : **addlw k**
هذه التعليمة تضيف (تجمع) قيمة ثابت محدد حرفيا literal والذي قيمته هي k والتي يجب على المبرمج تحديدها إلى القيمة المحفوظة في السجل W
والنتيجة تحفظ في السجل W فلا يوجد هنا اختيار .
وكما في التعليمة **addwf** فإن جميع خانات الحالة بسجل الحالة تتأثر بهذه التعليمة .

-5 هيكلة برنامج بلغة مجمع Assembler الميكرومراقب PIC 16F84

5-1 المعلومات و الإعدادات:

هذا الجزء من البرنامج خاص بالتعريف بهدف البرنامج و كذا صاحبه و تاريخ انجازه الخ...

ملاحظة: كل ما هو مكتوب بعد الفاصلة المنقوطة يعتبر من التعليقات و لا يِؤخذ بعين الاعتبار عند تحويل البرنامج (أي لا يحول إلى لغة الآلة).

;*

;   هذا البرنامج عبارة عن مثال لبرمجة الميكروماراقب بلغة المجمع

;

;*

; اسم البرنامج : التحكم في ثنائي ضوئي باستعمال زر ضاغط

; التاريخ : ....

;

;*

- التعريف بالميكرومراقب المستعمل ضروري للبرنامج الذي سيحول لغة المجمع إلى للغة الآلة و ذلك يتم عن طريق موجه (**directive**) لغة المجمع (**LIST**).
- كما يمكن إدراج ملف إضافي تعرف به مختلف الثوابت و الارفاقات (لتفادي كثافة البرنامج المكتوب)،

الموجة: < #include <p16F84A.inc

- **تهيئة كلمة الإعدادات** Configuration Word settings

الموجه **__CONFIG** يسمح للمبرمج بتعريف بعض خصائص وسمات الميكرومراقب عند تحميله بالبرنامج .وتكون هذه الخصائص ثابتة لحين برمجة الميكرومراقب مرة أخرى ، و هي تختلف عن الكلمات الأخرى حيث أنها تكون جزء من البرنامج ولكن لا يمكن الوصول إليها (كباقي الكلمات) سواء عن طريق البرنامج أو أي طريق آخر طالما أن البرنامج في حالة تشغيل. معاني هذه الكلمات موجودة بالملف **include**.
و هذا تفسير البعض منها.

;	_CP_ON	Protection Code: ON : ((لا يمكن قراءة محتوى الميكرومراقب
;	_CP_OFF	Protection Code: OFF
; (	_PWRTE_ON	Timer reset on power on: operational الخدمة في الطاقة توصيل عند المؤقت) reset
;	_PWRTE_OFF	Timer reset: out of operation
;	_WDT_ON	Watch-dog: operational(الخدمة في الحراسة مؤقت)
;	_WDT_OFF	Watch-dog: out of operation
;	_LP_OSC	Low speed quartz oscillator (مذبذب الكوارتز بسرعة عالية)
;	_XT_OSC	Medium speed quartz oscillator
;	_HS_OSC	High speed quartz oscillator
;	_RC_OSC	RC oscillator)RC(مذبذب بخلية

; *

LIST p=16F84A   ; تعريف بالميكرومراقب

#include <p16f84a.inc>   ; تعريف الثوابت الخاصة بالميكرومراقب

_CONFIG _CP_OFF & _WDT_OFF & _PWRTE_OFF & _XT_OSC   ; النظام تعريف ببعض خصائص

****

في هذه الحالة فإن الإعدادات هي كالتالي:
شفرة الحماية فعّالة، مؤقت الحراسة خارج الخدمة، reset المؤقت عند توصيل الطاقة خارج الخدمة، و المذبذب ذو الكوارتز بسرعة متوسطة هو المستعمل.

- الموجه **ORG** متبوع بعنوان، يحدد عنوان تنفيذ التعليمة الموالية و التي يجب أن تكون 0x0 لأن عند **reset** فإن البرنامج يعود إلى العنوان 0.
- التعليمة **GOTO** تأمر البرنامج بالقفز إلى عنوان الكلمة **init**

```
;**************************** إعادة التشغيل بعد reset **************************

        ORG     0x000           ;  عنوان الانطلاق بعد reset
        GOTO    init

**********************************************************************
```

5-2 تهيئة الميكرومراقب

يبدأ البرنامج في هذه المرحلة بتنفيذ تعليمات التهيئة وتسمى **initialization**

```
;**********************************************************************
;Initialisation
;**********************************************************************

init

        BSF         STATUS,5            ;وضع 1 في البت 5 لسجل الحالة للذهاب إلى البنك1;
        MOVLW           0x00            ; وضع 00000000 في سجل العمل W ;
        MOVWF           TRISB                   وضع محتوى سجل العمل فيB لبرمجته كمنفذ خروج ;
المنفذ
        MOVLW           0x1F            ; وضع 11111 في سجل العمل W ;
        MOVWF           TRISA                   وضع محتوى سجل العمل في A لبرمجته كمنفذ دخول;
المنفذ
        BCF             STATUS,5                نضع 0 في البت 5 لسجل الحالة للعودة إلى البنك0;
        GOTO        start               ; اقفز إلى البرنامج الرئيسي

;**********************************************************************
```

5-3 البرنامج الرئيسي

نصل إلى الجزء النهائي و هو البرنامج **الفعلي** نفسه.

```
;**********************************************************************
;     البرنامج الرئيسي
;**********************************************************************

start
        btfss       PORTA , 2       ;  افحص المنفذRA2واقفز إذا كان يساوي 1
        bsf         PORTB , 2       ; LED  ضع 1 على RB2 إذن تشتعل

        btfsc       PORTA , 2       ;  افحص المنفذRA2و اقفز إذا كان يساوي 0
        bcf         PORTB , 2       ; LED  ضع 0 على RB2 إذن تنطفئ

        goto        start           ; ارجع إلى start

        END                         ; موجه نهاية البرنامج

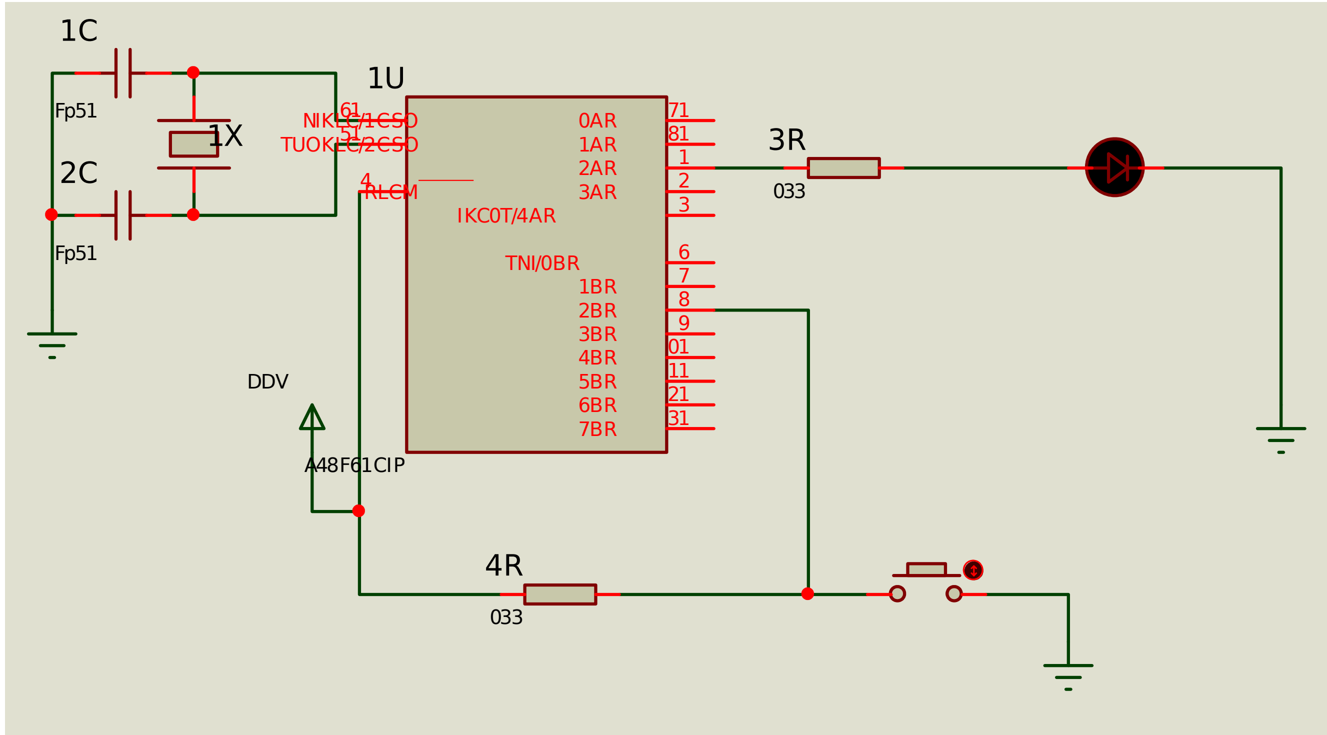
;**********************************************************************
```

البرنامج يفحص المدخل **RA2** عبر التعليمة **btfss** و يجري قفز على التعليمة المولية مادام **RA2** في المستوى المنطقي **1**. التعليمة **btfsc** تفحص بدورها المدخل **RA2** و تجري قفز على التعليمة المولية إذا كان المستوى المنطقي **0**. ثم يرجع إلى **start** . و إلّا تضع 0 على المخرج **RB2** (الثائي الضوئي يبقى منطفئ)، ثم بعد ذلك الرجوع إلى **start**.

و ريثما نضغط على الزر يأخذ **RA2** القيمة 0 فيمر البرنامج إلى التعليمة الموالية له مباشرة (**bsf PORTB , 2**) أي وضع **1** على المنفذ **RB2** و بالتالي يشتعل الثنائي الضوئي. ثم يعود البرنامج لفحص المدخل **RA2** من جديد، و ستمر تنفيذ البرنامج على هذا النحو .
و ينتهي البرنامج من خلال الموجه **END**.

6-   أنشطة:

أنجز البرامج بلغة المجمع حسب الدارة المبينة في الشكل التالي:



6-1 المشروع 1: التحكم في إشعال و إطفاء ثنائي ضوئي **LED** بواسطة زر ضاغط **SW** . كل ضغط متجدد على **SW** ينتج عنه قلب حالة الثنائي الضوئي.

6-2 المشروع 2: التحكم في إشعال ثنائي ضوئي **LED** بواسطة زر ضاغط **SW**. الضغط على SW ينتج عنه إشعال الثنائي الضوئي لمدة معينة **T= 5 s**)نفس الشكل السابق(.

ملاحظة: يجب انجاز برنامج فرعي للتأجيل باستعمال التعليمتان **CALL** و **RETURN**.

6-3 المشروع 3: الغمّازة (Clignotant)

الضغط على زر ضاغط SW ينتج عنه إشعال الثنائي الضوئي LED لمدة معينة **T= 2s** و إطفائه لمدة **T'= 2s** بصفة مستمرة)نفس الشكل السابق(.

```
;*********************************************************************
;                    SOUS-ROUTINE DE TEMPORISATION                          *
;*********************************************************************
;---------------------------------------------------------------------------------------------
; Cette sous-routine introduit un retard de 500.000 µs.
; Elle ne reçoit aucun paramètre et n'en retourne aucun
;---------------------------------------------------------------------------------------------
tempo
        movlw       2           ; pour 2 boucles
        movwf       cmpt3               ; initialiser compteur3
boucle3
        clrf    cmpt2           ; effacer compteur2
boucle2
```

```
        clrf    cmpt1               ; effacer compteur1
boucle1
        nop                     ; perdre 1 cycle *256 *256 *2
        decfsz        cmpt1 , f     ; décrémenter compteur1
        goto  boucle1       ; si pas 0, boucler
        decfsz        cmpt2 , f     ; si 0, décrémenter compteur 2
        goto  boucle2       ; si cmpt2 pas 0, recommencer boucle1
        decfsz        cmpt3 , f     ; si 0, décrémenter compteur 3
        goto  boucle3       ; si cmpt3 pas 0, recommencer boucle2
      return                                            ; retour de la
                                                              sous-routine
```

تمرين2

b) Fichier à extension .asm

```
;Titre du programme : PROG1
;Ce programme allume la LED branchée sur la
;sortie RB0 (bit 0 du Port B) et la laisse
;indéfiniment allumée.
;+++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
; DIRECTIVES
;+++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
PROCESSOR 16F84
RADIX HEX
INCLUDE « P16F84.INC »
__CONFIG 3FF1
;+++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
; VECTEUR de RESET
;+++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
ORG 00 ;Vecteur de Reset.
GOTO START ;Renvoi à l'adresse EEPROM 05 (hexa)
;+++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
; INITIALISATIONS
;+++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
START ORG 05 ;Saut introduit pour passer au-dessus
;des 5 premières adresses de la mémoire
;EEPROM (00 – 01 – 02 – 03 et 04).
CLRF PORTB ;Initialise le Port B.
BSF STATUS,RP0 ;Met à 1 (set) le bit 5 (RP0) du
;registre d'état (STATUS).
;Autrement dit : sélectionne la
;page 1 du Register File (adresses
;de 80 à 8B) dans laquelle se trouve
;le Registre STATUS (à l'adresse 83).
MOVLW b'00000000' ;Met la valeur binaire 00000000 dans
;le registre W, matérialisant ainsi notre
;intention d'utiliser les 8 lignes du
;Port B comme SORTIES.
```

```
MOVWF TRISB ;Port B configuré, mais encore en
;haute impédance (Trhee-state).
.BCF STATUS,RP0 ;Retour à la page 0 du Register File



;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; PROGRAMME
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++
LOOP BSF PORTB,0 ;Allume la LED, car l'instruction
; « BSF » met à 1 (set).
;Dans le cas présent, elle met à 1 le
;bit zéro du Port B (PORTB,0).
GOTO LOOP ;Le programme se reboucle.
;La LED reste indéfiniment allumée.
.END ;Fin du programme
```

تمرين3

b) Fichier à extension .asm

```
;Titre du programme : PROG2
;Ce programme fait clignoter indéfiniment la LED branchée sur la
;sortie RB0 (bit 0 du Port B).
;Le programme comporte une temporisation (DELAI) pour rendre
;perceptibles les allumages et les extinctions de la LED, sinon les
;transitions auraient lieu à très grande vitesse et notre oeil ne verrait
;pas les clignotements.
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
; DIRECTIVES
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
PROCESSOR 16F84
RADIX HEX
INCLUDE « P16F84.INC »
__CONFIG 3FF1
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
; DECLARATIONS DES VARIABLES
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
COMPT1 EQU 0C ;On met la variable COMPT1 à
;l'adresse RAM 0C.
COMPT2 EQU 0D ;On met la variable COMPT2 à
;l'adresse RAM 0D.
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
; VECTEUR DE RESET
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
ORG 00 ;Vecteur de Reset.
GOTO START ;Renvoi à l'adresse EEPROM 05 (hexa)
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
```

```
; INITIALISATIONS
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
START ORG 05 ;Saut introduit intentionnellement pour faire
;démarrer le programme à l'adresse EEPROM 05.
CLRF PORTB ;Efface les 8 bits du Port B.
BSF STATUS,RP0 ;Met à 1 (set) le bit 5 (RP0) du
;registre d'état (STATUS).
;Autrement dit : sélectionne la
;page 1 du Register File (adresses
;de 80 à 8B) dans laquelle se trouve
;le Registre STATUS (à l'adresse 83).
MOVLW b'00000000' ;Met la valeur binaire 00000000 dans
;le registre W, matérialisant ainsi notre
intention d'utiliser les 8 lignes;
;Port B comme SORTIES.
;La notation b'00000000' indique que
;la valeur 00000000 est à interpréter
;en tant que chiffre binaire.
MOVWF TRISB ;Port B configuré, mais encore en
;haute impédance (Trhee-state).
BCF STATUS,RP0 ;Retour à la page 0 du Register File.
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++
; PROGRAMME PRINCIPAL
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++
MAIN BCF PORTB,0 ;LED éteinte car l'instruction
; « BCF » met à 0 (clear).
;Ici, elle met à 0 le bit 0 du
;Port B (PORTB,0).
CALL DELAI ;Appelle le sous-programme de
;retard (DELAI).
BSF PORTB,0 ;LED allumée, car l'instruction
; « BSF » met à 1 (set).
;Ici elle met à 1 le bit 0 du
;Port B (PORTB,0).
CALL DELAI ;On appelle à nouveau le
;sous-programme de retard.
GOTO MAIN ;Retour au programme principal.
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++
; SOUS-PROGRAMME de TEMPORISATION
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++
DELAI DECFSZ COMPT1,1           ;Décrémente COMPT1et - s'il n'est pas
GOTO DELAI                      ;à zéro - va à DELAI
MOVLW .255                      ;Charge la variable COMPT1 (adresse
MOVWF COMPT1                    ;RAM 0C) avec 255 (en décimal).
DECFSZ COMPT2,1 ;Décrémente COMPT2, et s'il n'est pas
GOTO DELAI ;à zéro, va à DELAI
MOVLW .255 ;Recharge COMPT1 avec 255
MOVWF COMPT1
MOVLW .255 ;Recharge COMPT2 avec 255
```

```
MOVWF COMPT2
RETURN ;Fin du sous-programme DELAI
.END ;Fin du programme
```